WHAT IS CLAIMED:

1. A method of performing native binding to execute native code during the translation of subject program code executable by a subject processor to target program code executable by a target processor, wherein native code is code executable by the target processor, said method comprising:

identifying certain subject program code having corresponding native code;

identifying the native code which corresponds to the identified subject program code; and

executing the corresponding native code instead of executing a translated version of the identified subject program code.

2. The method of claim 1, wherein the identified subject program code corresponds to a subject function and the identified native code corresponds to a native function, wherein the native code executing step comprises:

executing the native function instead of the subject function in the translation of the subject program code.

3. The method of claim 2, wherein the native function executing step comprises:

transforming zero or more function parameters from a target code representation to a native code representation;

invoking the native function with the transformed function parameter according to a prototype of the native function; and transforming zero or more return values of the invoked native function from a native code representation to a target code representation.

4. The method of claim 3, wherein at least one of the transformations in the transforming steps generates an intermediate representation of the transformation.

5. The method of claim 3, wherein at least one of the transformations in the transforming steps generates target code.

6. The method of claim 3, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers according to a uniform call stub interface; and

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

7. The method of claim 3, wherein the native function executing step comprises:

transforming a function parameter from a target code representation to a native code representation;

invoking the native function with the transformed function parameter according to a prototype of the native function; and transforming a result of the invoked native function from a native code representation to a target code representation.

8. The method of claim 3, wherein the function parameter transforming step and the native function invoking step are described in subject code by translator specific instructions added to the subject instruction set.

9. The method of claim 1, wherein the steps of identifying the certain subject code and its corresponding native code are performed using a bind point description.

10. The method of claim 9, wherein the bind point description includes a subject function and a native function, wherein the subject function identifies the certain subject program code having corresponding native code and the native function identifies the corresponding native code.

11. The method of claim 10, further comprising inserting in the target code a call stub to the native function during translation of the subject code when encountering the subject function contained in the bind point description.

12. The method of claim 9, wherein the bind point description is embedded within a translator performing the translation.

13. The method of claim 9, further comprising reading the bind point description from a stored bind point description file at the beginning of translation execution.

14. The method of claim 9, wherein the bind point description includes a location in the subject code and a corresponding native function, wherein the location in the subject code identifies the certain subject program code having corresponding native code and the native function identifies the corresponding native code.

15. The method of claim 9, wherein the bind point description includes a location in the subject code and a reference to code to be invoked, wherein the location in the subject code identifies the certain subject program code having corresponding native code and the reference to code to be invoked identifies the corresponding native code.

16. The method of claim 15, wherein the code to be invoked is target code.

17. The method of claim 9, wherein the bind point description includes a native function call which is inserted in the target code either before, after, or in place of a subject function call.

18. The method of claim 9, further performing runtime symbol patching comprising:

encoding subject-to-native function mappings in a symbol table of the subject program,

replacing entries in the symbol table of the subject program with special native binding markers, and interpreting the special native binding markers when encountered during translation as bind point descriptions to identify an appropriate native function to call.

19. The method of claim 9, wherein the bind point description includes a correspondence to an external Schizo call command, wherein the Schizo call command is a translator-specific native binding instruction, the method comprising:

when encountering a bind point description identifying an external Schizo call command during translation of the subject code, diverting the flow of translation to the execution of the external Schizo call command.

20. The method of claim 19, wherein the external Schizo call command execution step comprises:

interpreting the external Schizo call command; and

generating an intermediate representation of the external Schizo call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

21. The method of claim 19, wherein the external Schizo call command execution step comprises:

interpreting the external Schizo call command; and

generating target code for the external Schizo call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

22. The method of claim 1, further comprising:

inserting Schizo call commands into the subject code, wherein Schizo call commands are translator-specific native binding instructions; and

detecting the Schizo call commands during translation of the subject code.

23. The method of claim 22, further comprising:

when encountering a Schizo call command during translation of the subject code, diverting the flow of translation to the execution of the Schizo call command.

24. The method of claim 23, wherein the Schizo call command execution step comprises:

interpreting the external Schizo call command; and

generating an intermediate representation of the Schizo call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

25. The method of claim 23, wherein the Schizo call command execution step comprises:

interpreting the Schizo call command; and

generating target code for the Schizo call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function

26. The method of claim 22, wherein the Schizo call commands are variable length instructions including multiple sub-component instructions.

27. The method of claim 26, wherein the multiple sub-component instructions include a Schizo Escape sub-component instruction, said Schizo call commands detecting step further comprising detecting the Schizo Escape sub-component instruction.

28. The method of claim 27, wherein said Schizo Escape sub-component instruction further identifies a type of Schizo call command represented by the other sub-component instructions of the Schizo call command.

29. The method of claim 1, further comprising:

parsing and decoding a native binding implementation scripting language containing native binding scripts;

interpreting the native binding scripts during translation;

generating an intermediate representation of the native binding scripts to transform a function parameter from a target code representation to a native code representation.

30. The method of claim 29, further comprising:

integrating the intermediate representation of the native binding scripts into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

31. The method of claim 1, further comprising:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers according to a uniform call stub interface;

interpreting the native code call stub function; and

generating an intermediate representation of the native code call stub function binding scripts to transform a function parameter from a target code representation to a native code representation.

32. The method of claim 21, further comprising:

integrating the intermediate representation of the native code call stub function into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

33. The method of claim 3, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers; and

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

34. The method of claim 1, further comprising:

parsing a scripting language implementation of a native code call stub function;

compiling the parsed native code call stub function into a native code executable module; and

linking the native code executable module with an executable for performing the translation.

35. The method of claim 34, wherein the native code executable module is executable for:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers; and

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

36. The method of claim 34, wherein the steps of identifying the certain subject code and its corresponding native code are performed using a bind point description, said bind point description including a subject function and a native code call stub function, wherein the subject function identifies the certain subject program code having corresponding native code and the native code call stub function identifies the corresponding native code.

37. The method of claim 36, further comprising encoding the identity of the native function of the native code call stub function in the scripting language implementation of the native code executable module.

38. The method of claim 3, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a target code call stub function with the transformed subject registers; and

invoking from the target code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

39. The method of claim 38, further comprising:

generating an intermediate representation of the native function executing step;

integrating the intermediate representation of the native function executing step into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

40. The method of claim 1, wherein the subject function to be executed is a system call.

41. The method of claim 1, wherein the subject function to be executed is a library function.

42. A computer-readable storage medium having software resident thereon in the form of computer-readable code executable by a computer to perform the following native binding steps to execute native code during the translation of subject program code executable by a subject processor to target program code executable by a target processor, wherein native code is code executable by the target processor, said steps comprising:

identifying certain subject program code having corresponding native code;

identifying the native code which corresponds to the identified subject program code; and executing the corresponding native code instead of executing a translated version of the identified subject program code.

43. The computer-readable storage medium of claim 42, wherein the identified subject program code corresponds to a subject function and the identified native code corresponds to a native function, wherein the native code executing step comprises:

executing the native function instead of the subject function in the translation of the subject program code.

44. The computer-readable storage medium of claim 43, wherein the native function executing step comprises:

transforming zero or more function parameters from a target code representation to a native code representation;

invoking the native function with the transformed function parameter according to a prototype of the native function; and

transforming zero or more return values of the invoked native function from a native code representation to a target code representation.

45. The computer-readable storage medium of claim 44, wherein at least one of the transformations in the transforming steps generates an intermediate representation of the transformation.

46. The computer-readable storage medium of claim 44, wherein at least one of the transformations in the transforming steps generates target code.

47. The computer-readable storage medium of claim 44, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers according to a uniform call stub interface; and

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

48. The computer-readable storage medium of claim 44, wherein the native function executing step comprises:

transforming a function parameter from a target code representation to a native code representation;

invoking the native function with the transformed function parameter according to a prototype of the native function; and

transforming a result of the invoked native function from a native code representation to a target code representation.

49. The computer-readable storage medium of claim 44, wherein the function parameter transforming step and the native function invoking step are described in subject code by translator specific instructions added to the subject instruction set.

50. The computer-readable storage medium of claim 42, wherein the steps of identifying the certain subject code and its corresponding native code are performed using a bind point description.

51. The computer-readable storage medium of claim 50, wherein the bind point description includes a subject function and a native function, wherein the subject function identifies the certain subject program code having corresponding native code and the native function identifies the corresponding native code.

52. The computer-readable storage medium of claim 51, said computer-readable code executable further executable for inserting in the target code a call stub to the native function during translation of the subject code when encountering the subject function contained in the bind point description.

53. The computer-readable storage medium of claim 50, wherein the bind point description is embedded within a translator performing the translation.

54. The computer-readable storage medium of claim 50, said computer-readable code executable further executable for reading the bind point description from a stored bind point description file at the beginning of translation execution.

55. The computer-readable storage medium of claim 50, wherein the bind point description includes a location in the subject code and a corresponding native function, wherein the location in the subject code identifies the certain subject program

code having corresponding native code and the native function identifies the corresponding native code.

56. The computer-readable storage medium of claim 50, wherein the bind point description includes a location in the subject code and a reference to code to be invoked, wherein the location in the subject code identifies the certain subject program code having corresponding native code and the reference to code to be invoked identifies the corresponding native code.

57. The computer-readable storage medium of claim 56, wherein the code to be invoked is target code.

58. The computer-readable storage medium of claim 50, wherein the bind point description includes a native function call which is inserted in the target code either before, after, or in place of a subject function call.

59. The computer-readable storage medium of claim 50, said computer-readable code executable further executable for performing runtime symbol patching comprising:

encoding subject-to-native function mappings in a symbol table of the subject program,

replacing entries in the symbol table of the subject program with special native binding markers, and interpreting the special native binding markers when encountered during translation as bind point descriptions to identify an appropriate native function to call.

60. The computer-readable storage medium of claim 50, wherein the bind point description includes a correspondence to an external Schizo call command, wherein the Schizo call command is a translator-specific native binding instruction, said computer-readable code executable further executable for:

when encountering a bind point description identifying an external Schizo call command during translation of the subject code, diverting the flow of translation to the execution of the external Schizo call command.

61. The computer-readable storage medium of claim 60, wherein the external Schizo call command execution step comprises:

interpreting the external Schizo call command; and

generating an intermediate representation of the external Schizo call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

62. The computer-readable storage medium of claim 60, wherein the external Schizo call command execution step comprises:

interpreting the external Schizo call command; and generating target code for the external Schizo call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

63. The computer-readable storage medium of claim 42, said computer-readable code executable further executable for performing the following steps:

inserting Schizo call commands into the subject code, wherein Schizo call commands are translator-specific native binding instructions; and

detecting the Schizo call commands during translation of the subject code.

64. The computer-readable storage medium of claim 63, said computer-readable code executable further executable for performing the following steps:

when encountering a Schizo call command during translation of the subject code, diverting the flow of translation to the execution of the Schizo call command.

65. The computer-readable storage medium of claim 64, wherein the Schizo call command execution step comprises:

interpreting the external Schizo call command; and

generating an intermediate representation of the Schizo call command which:

transforms a function parameter from a target code representation to a native code representation, and invokes the native function with the transformed function parameter according to a prototype of the native function.

66. The computer-readable storage medium of claim 64, wherein the Schizo call command execution step comprises:

interpreting the Schizo call command; and

generating target code for the Schizo call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function

67. The computer-readable storage medium of claim 63, wherein the Schizo call commands are variable length instructions including multiple sub-component instructions.

68. The computer-readable storage medium of claim 67, wherein the multiple sub-component instructions include a Schizo Escape sub-component instruction, said Schizo call commands detecting step further comprising detecting the Schizo Escape sub-component instruction.

69. The computer-readable storage medium of claim 68, wherein said Schizo Escape sub-component instruction further identifies a type of Schizo call command represented by the other sub-component instructions of the Schizo call command.

70. The computer-readable storage medium of claim 42, said computer-readable code executable further executable for performing the following steps:

parsing and decoding a native binding implementation scripting language containing native binding scripts;

interpreting the native binding scripts during translation; and

generating an intermediate representation of the native binding scripts to transform a function parameter from a target code representation to a native code representation.

71. The computer-readable storage medium of claim 70, said computer-readable code executable further executable for performing the following steps:

integrating the intermediate representation of the native binding scripts into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

72. The computer-readable storage medium of claim 42, said computer-readable code executable further executable for performing the following steps:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers according to a uniform call stub interface;

interpreting the native code call stub function; and generating an intermediate representation of the native code call stub function binding scripts to transform a function parameter from a target code representation to a native code representation.

73. The computer-readable storage medium of claim 62, said computer-readable code executable further executable for performing the following steps:

integrating the intermediate representation of the native code call stub function into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest

74. The computer-readable storage medium of claim 44, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers; and

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

75. The computer-readable storage medium of claim 42, said computer-readable code executable further executable for performing the following steps:

parsing a scripting language implementation of a native code call stub function;

compiling the parsed native code call stub function into a native code executable module; and

linking the native code executable module with an executable for performing the translation.

76. The computer-readable storage medium of claim 75, wherein the native code executable module is executable for:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers; and

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

77. The computer-readable storage medium of claim 75, wherein the steps of identifying the certain subject code and its corresponding native code are performed using a bind point description, said bind point description including a subject function and a native code call stub function, wherein the subject function identifies the certain subject program code having corresponding native code and the native code call stub function identifies the corresponding native code.

78. The computer-readable storage medium of claim 77, said computer-readable code executable further executable for encoding the identity of the native

function of the native code call stub function in the scripting language implementation of the native code executable module.

79. The computer-readable storage medium of claim 44, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a target code call stub function with the transformed subject registers; and

invoking from the target code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

80. The computer-readable storage medium of claim 79, said computer-readable code executable further executable for performing the following steps:

generating an intermediate representation of the native function executing step;

integrating the intermediate representation of the native function executing step into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

81. The computer-readable storage medium of claim 42, wherein the subject function to be executed is a system call.

82. The computer-readable storage medium of claim 42, wherein the subject function to be executed is a library function.

83. In combination:

a target processor; and

translator code for performing native binding to execute native code during the translation of subject program code executable by a subject processor to target program code executable by a target processor, wherein native code is code executable by the target processor, said translator code comprising code executable by said target processor for performing the following steps:

identifying certain subject program code having corresponding native code;

identifying the native code which corresponds to the identified subject program code; and

executing the corresponding native code instead of executing a translated version of the identified subject program code.

84. The combination of claim 83, wherein the identified subject program code corresponds to a subject function and the identified native code corresponds to a native function, wherein the native code executing step comprises:

executing the native function instead of the subject function in the translation of the subject program code.

85. The combination of claim 84, wherein the native function executing step comprises:

transforming zero or more function parameters from a target code representation to a native code representation;

invoking the native function with the transformed function parameter according to a prototype of the native function; and

transforming zero or more return values of the invoked native function from a native code representation to a target code representation.

86. The combination of claim 85, wherein at least one of the transformations in the transforming steps generates an intermediate representation of the transformation.

87. The combination of claim 85, wherein at least one of the transformations in the transforming steps generates target code.

88. The combination of claim 85, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers according to a uniform call stub interface; and

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

89. The combination of claim 85, wherein the native function executing step comprises:

transforming a function parameter from a target code representation to a native code representation;

invoking the native function with the transformed function parameter according to a prototype of the native function; and

transforming a result of the invoked native function from a native code representation to a target code representation.

90. The combination of claim 85, wherein the function parameter transforming step and the native function invoking step are described in subject code by translator specific instructions added to the subject instruction set.

91. The combination of claim 83, wherein the steps of identifying the certain subject code and its corresponding native code are performed using a bind point description.

92. The combination of claim 91, wherein the bind point description includes a subject function and a native function, wherein the subject function identifies the certain subject program code having corresponding native code and the native function identifies the corresponding native code.

93. The combination of claim 92, said translator code further comprising code executable by said target processor for inserting in the target code a call stub to the native

function during translation of the subject code when encountering the subject function contained in the bind point description.

94. The combination of claim 91, wherein the bind point description is embedded within a translator performing the translation.

95. The combination of claim 91, said translator code further comprising code executable by said target processor for reading the bind point description from a stored bind point description file at the beginning of translation execution.

96. The combination of claim 91, wherein the bind point description includes a location in the subject code and a corresponding native function, wherein the location in the subject code identifies the certain subject program code having corresponding native code and the native function identifies the corresponding native code.

97. The combination of claim 91, wherein the bind point description includes a location in the subject code and a reference to code to be invoked, wherein the location in the subject code identifies the certain subject program code having corresponding native code and the reference to code to be invoked identifies the corresponding native code.

98. The combination of claim 97, wherein the code to be invoked is target code.

99. The combination of claim 91, wherein the bind point description includes a native function call which is inserted in the target code either before, after, or in place of a subject function call.

100. The combination of claim 91, said translator code further comprising code executable by said target processor for performing runtime symbol patching comprising:

encoding subject-to-native function mappings in a symbol table of the subject program,

replacing entries in the symbol table of the subject program with special native binding markers, and

interpreting the special native binding markers when encountered during translation as bind point descriptions to identify an appropriate native function to call.

101. The combination of claim 91, wherein the bind point description includes a correspondence to an external Schizo call command, wherein the Schizo call command is a translator-specific native binding instruction, the method comprising:

when encountering a bind point description identifying an external Schizo call command during translation of the subject code, diverting the flow of translation to the execution of the external Schizo call command.

102. The combination of claim 101, wherein the external Schizo call command execution step comprises:

interpreting the external Schizo call command; and generating an intermediate representation of the external Schizo call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

103. The combination of claim 101, wherein the external Schizo call command execution step comprises:

interpreting the external Schizo call command; and

generating target code for the external Schizo call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

104. The combination of claim 83, said translator code further comprising code executable by said target processor for performing the following steps:

inserting Schizo call commands into the subject code, wherein Schizo call commands are translator-specific native binding instructions; and

detecting the Schizo call commands during translation of the subject code.

105. The combination of claim 104, said translator code further comprising code executable by said target processor for performing the following steps:

when encountering a Schizo call command during translation of the subject code, diverting the flow of translation to the execution of the Schizo call command.

106. The combination of claim 105, wherein the Schizo call command execution step comprises:

interpreting the external Schizo call command; and

generating an intermediate representation of the Schizo call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

107. The combination of claim 105, wherein the Schizo call command execution step comprises:

interpreting the Schizo call command; and

generating target code for the Schizo call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

108. The combination of claim 104, wherein the Schizo call commands are variable length instructions including multiple sub-component instructions.

109. The combination of claim 108, wherein the multiple sub-component instructions include a Schizo Escape sub-component instruction, said Schizo call commands detecting step further comprising detecting the Schizo Escape sub-component instruction.

110. The combination of claim 109, wherein said Schizo Escape sub-component instruction further identifies a type of Schizo call command represented by the other sub-component instructions of the Schizo call command.

111. The combination of claim 83, said translator code further comprising code executable by said target processor for performing the following steps:

parsing and decoding a native binding implementation scripting language containing native binding scripts;

interpreting the native binding scripts during translation; and

generating an intermediate representation of the native binding scripts to transform a function parameter from a target code representation to a native code representation.

112. The combination of claim 111, said translator code further comprising code executable by said target processor for performing the following steps:

integrating the intermediate representation of the native binding scripts into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

113. The combination of claim 83, said translator code further comprising code executable by said target processor for performing the following steps:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers according to a uniform call stub interface;

interpreting the native code call stub function; and

generating an intermediate representation of the native code call stub function binding scripts to transform a function parameter from a target code representation to a native code representation.

114. The combination of claim 103, said translator code further comprising code executable by said target processor for performing the following steps:

integrating the intermediate representation of the native code call stub function into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

115. The combination of claim 85, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers;

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

116. The combination of claim 83, said translator code further comprising code executable by said target processor for performing the following steps:

parsing a scripting language implementation of a native code call stub function;

compiling the parsed native code call stub function into a native code executable module; and

linking the native code executable module with an executable for performing the translation.

117. The combination of claim 116, wherein the native code executable module is executable for:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers; and

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

118. The combination of claim 116, wherein the steps of identifying the certain subject code and its corresponding native code are performed using a bind point

description, said bind point description including a subject function and a native code call stub function, wherein the subject function identifies the certain subject program code having corresponding native code and the native code call stub function identifies the corresponding native code.

119. The combination of claim 118, said translator code further comprising code executable by said target processor for encoding the identity of the native function of the native code call stub function in the scripting language implementation of the native code executable module.

120. The combination of claim 85, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a target code call stub function with the transformed subject registers; and

invoking from the target code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

121. The combination of claim 120, said translator code further comprising code executable by said target processor for performing the following steps:

generating an intermediate representation of the native function executing step;

integrating the intermediate representation of the native function executing step into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

122. The combination of claim 83, wherein the subject function to be executed is a system call.

123. The combination of claim 83, wherein the subject function to be executed is a library function.